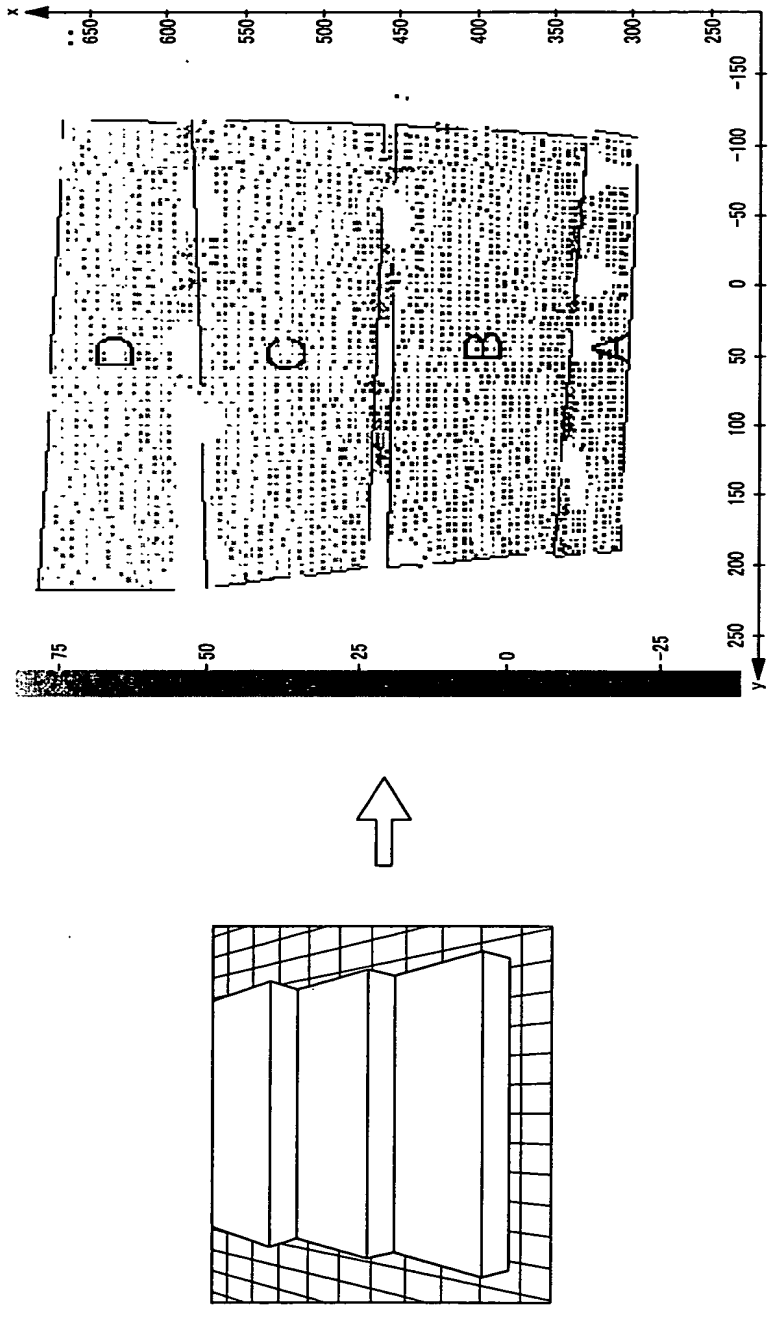



FIG.1

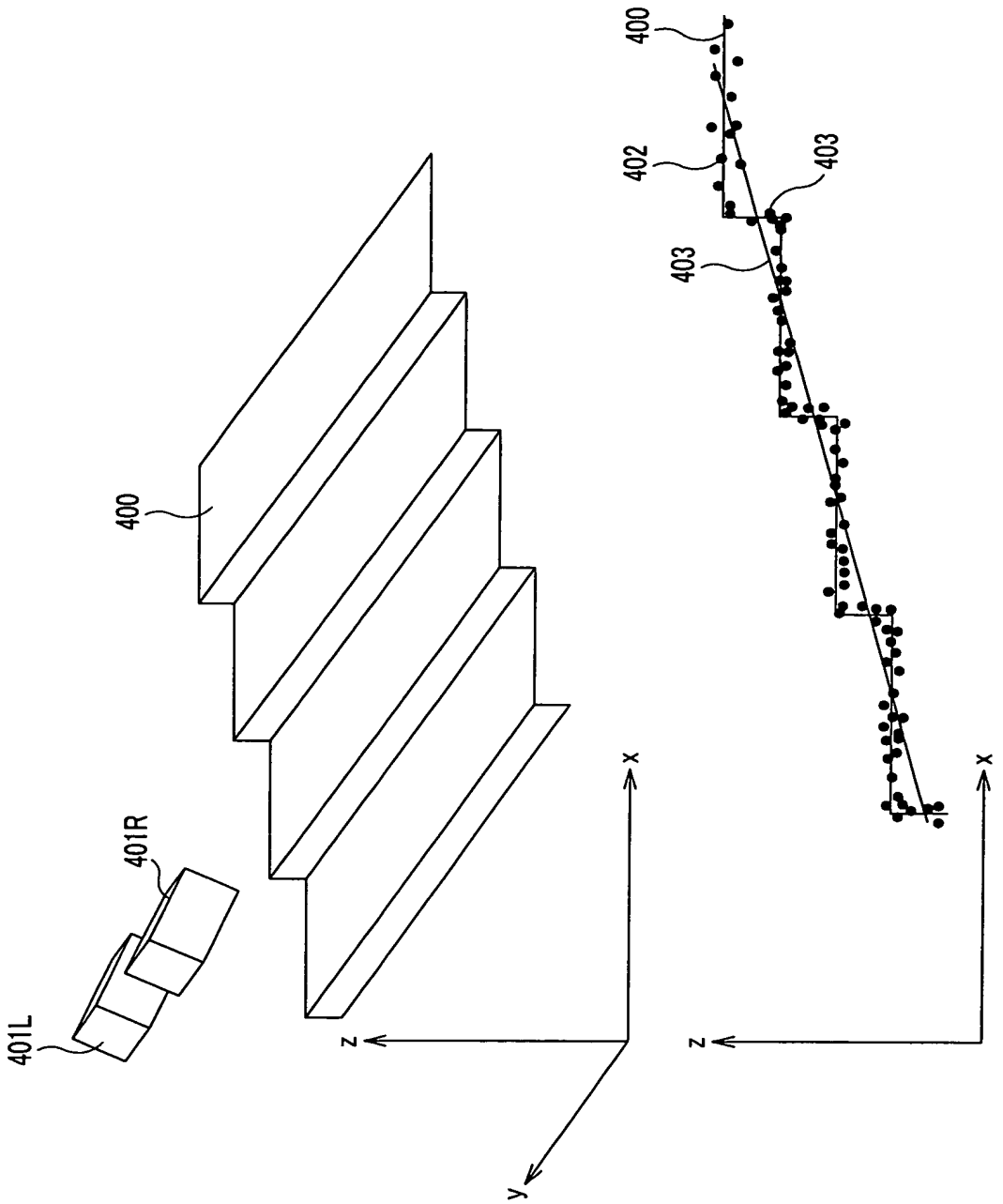



FIG.2

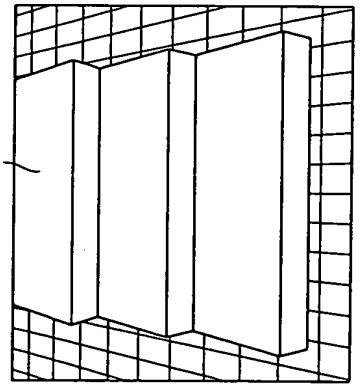



FIG.3A

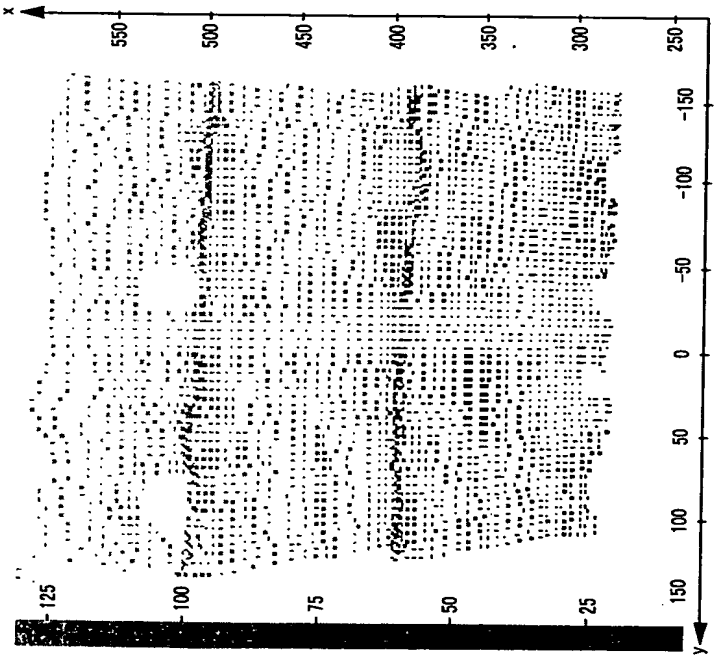



FIG.3B

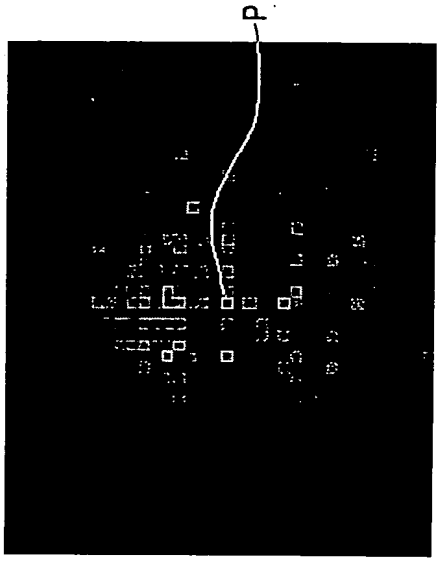



FIG.3C

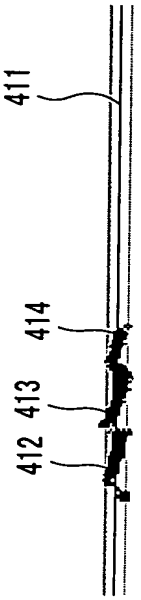



FIG.3D

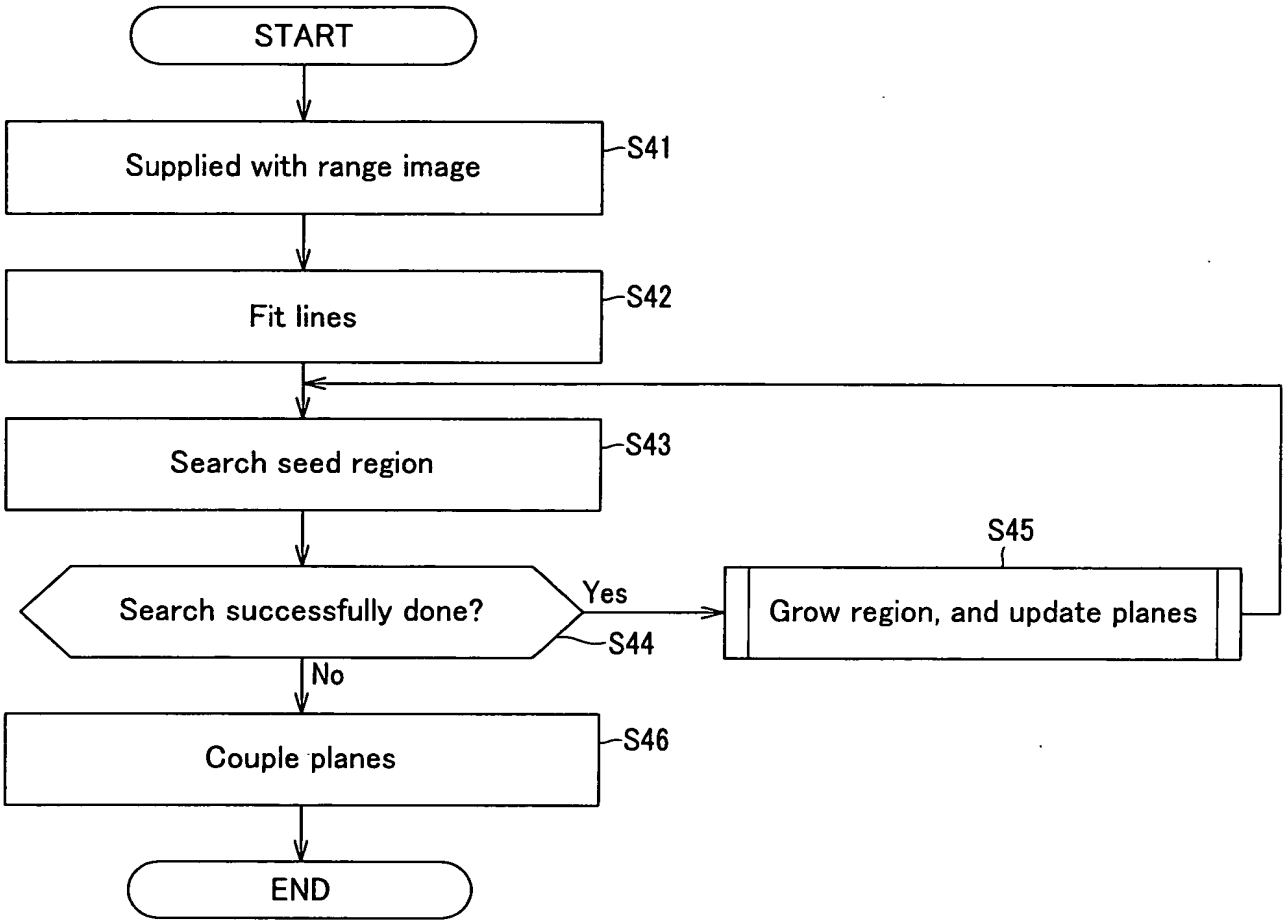

START
Supplied with range image
S41
Fit lines
S42
Search seed region
S43
Search successfully done?
Yes
S44
No
S45
Grow region, and update planes
Couple planes
S46
END

FIG.4

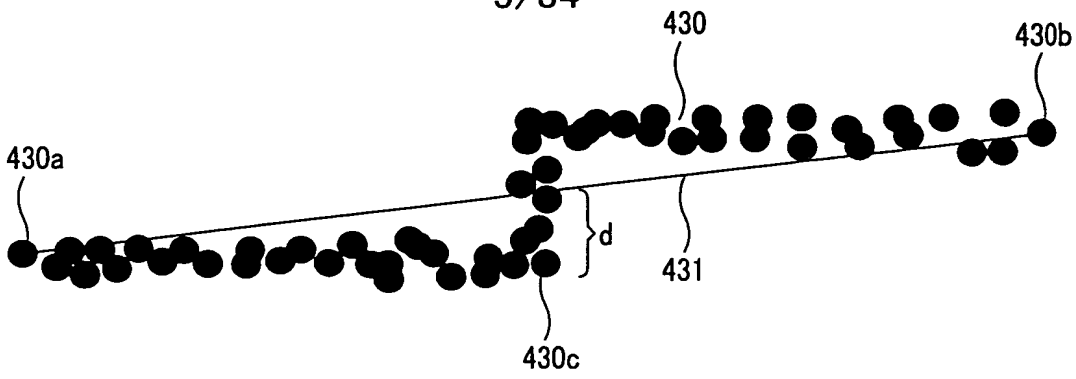



FIG.5A

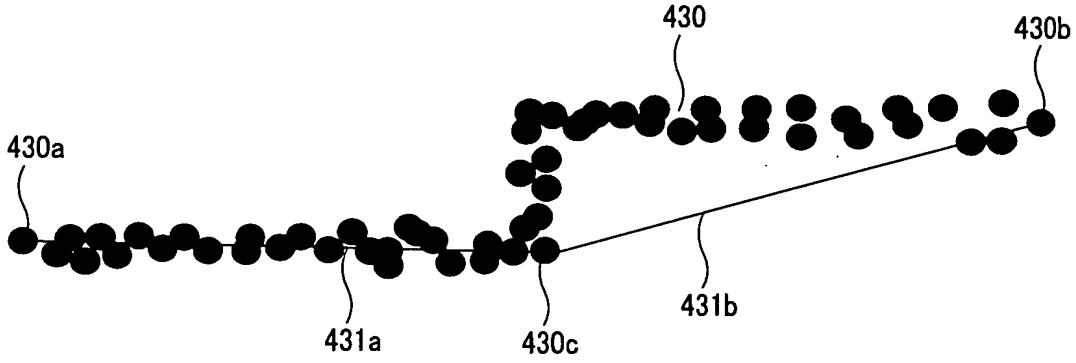



FIG.5B

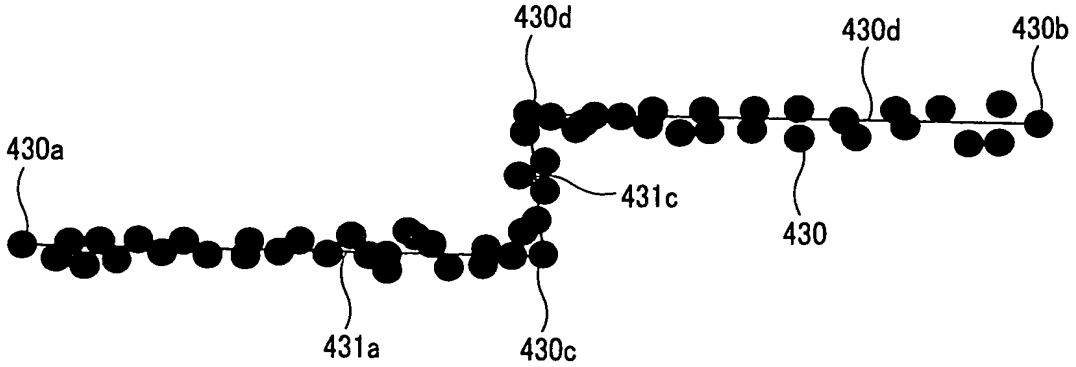



FIG.5C

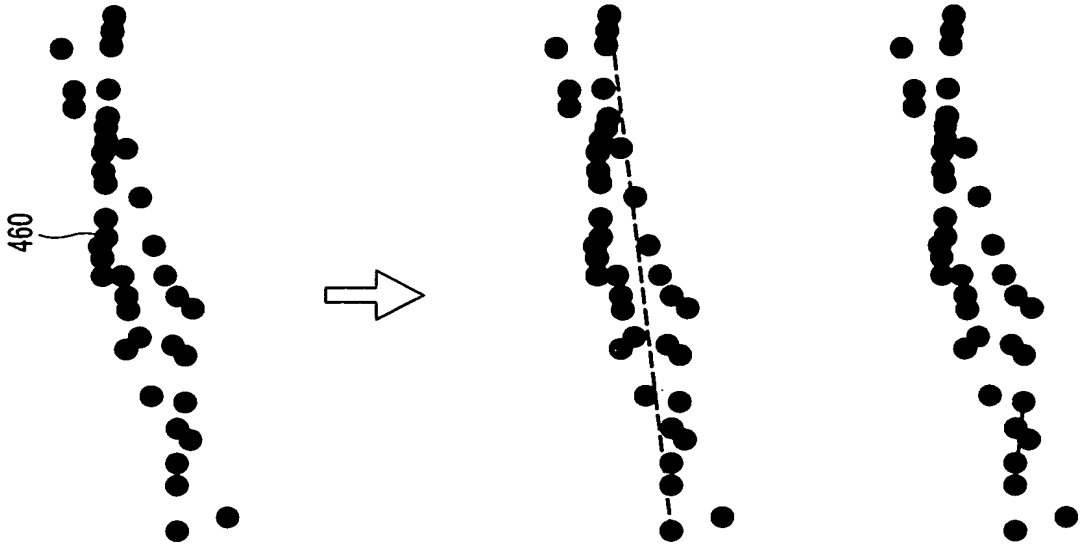



FIG.6B

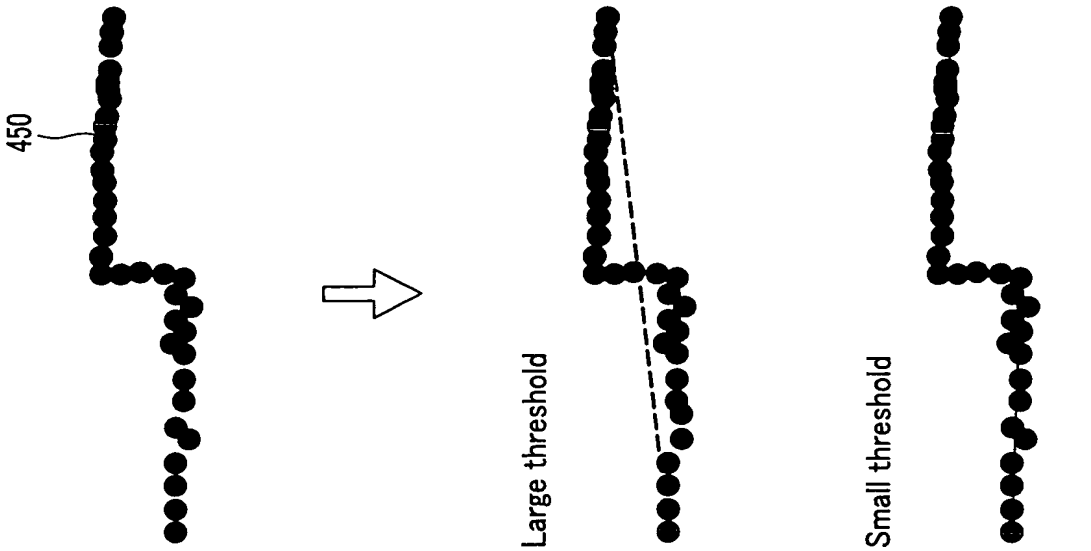



FIG.6A

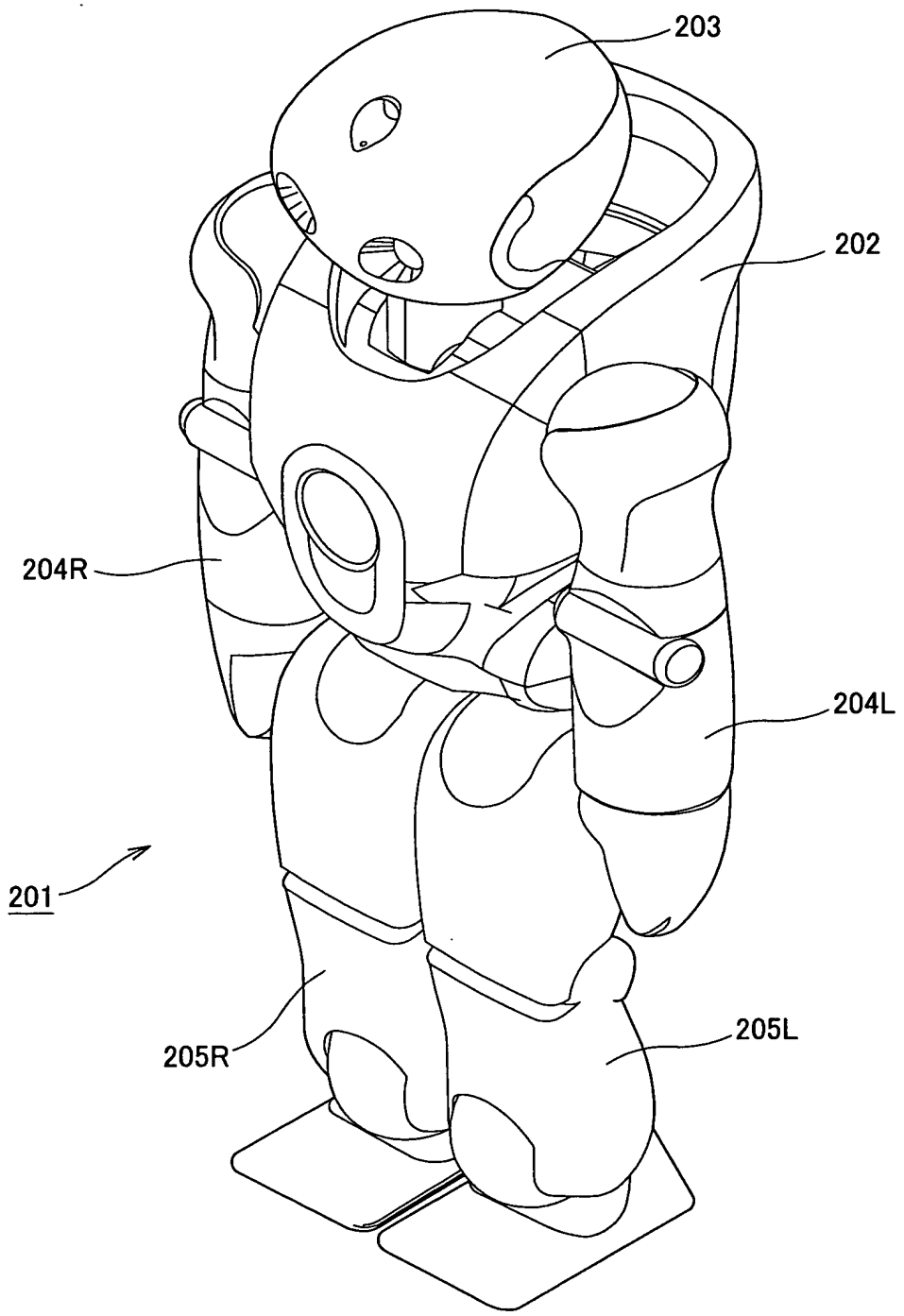



FIG.7

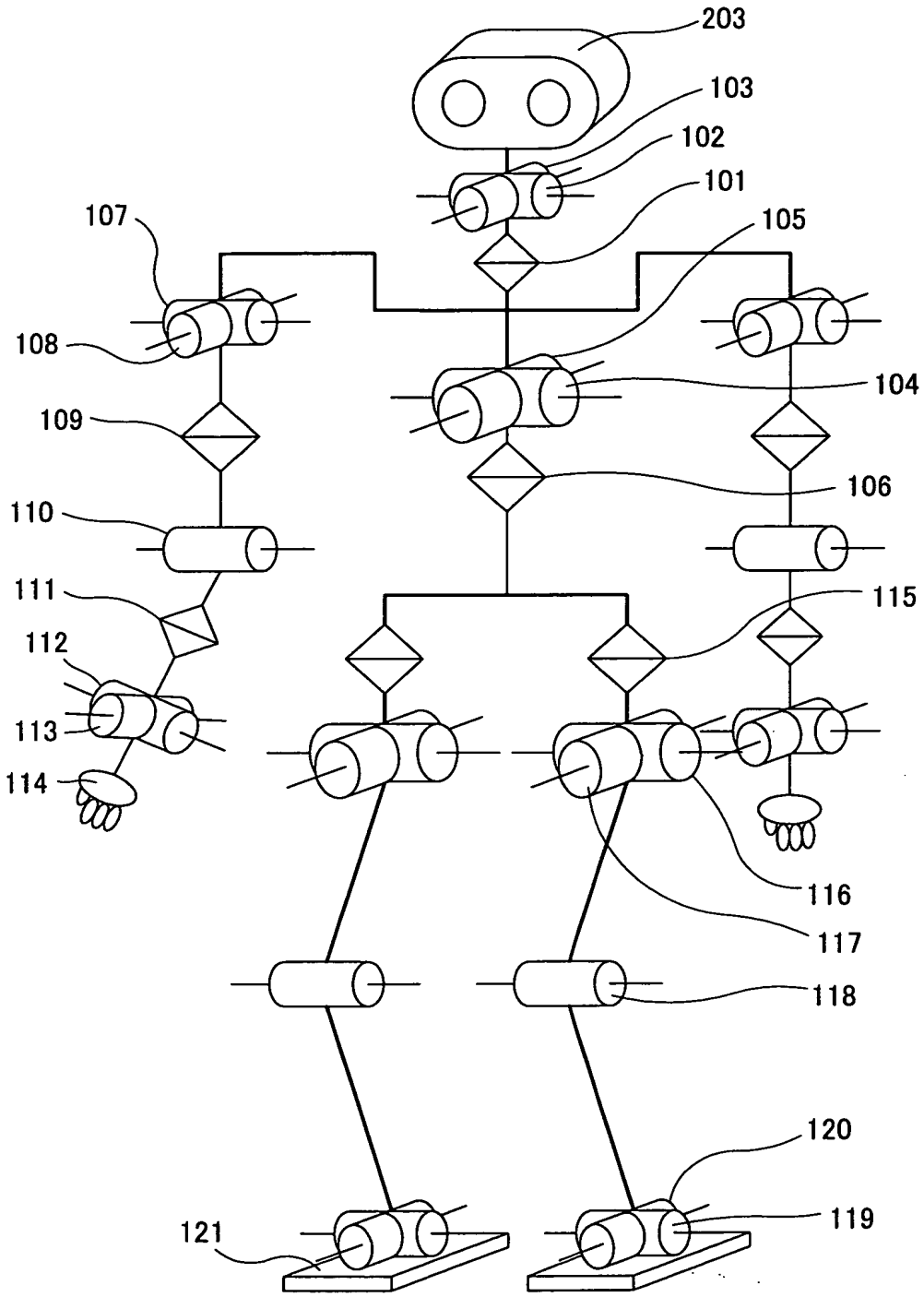

203
103
102
101
107
105
108
104
109
106
110
111
115
112
113
114
116
117
118
120
121
119

FIG.8

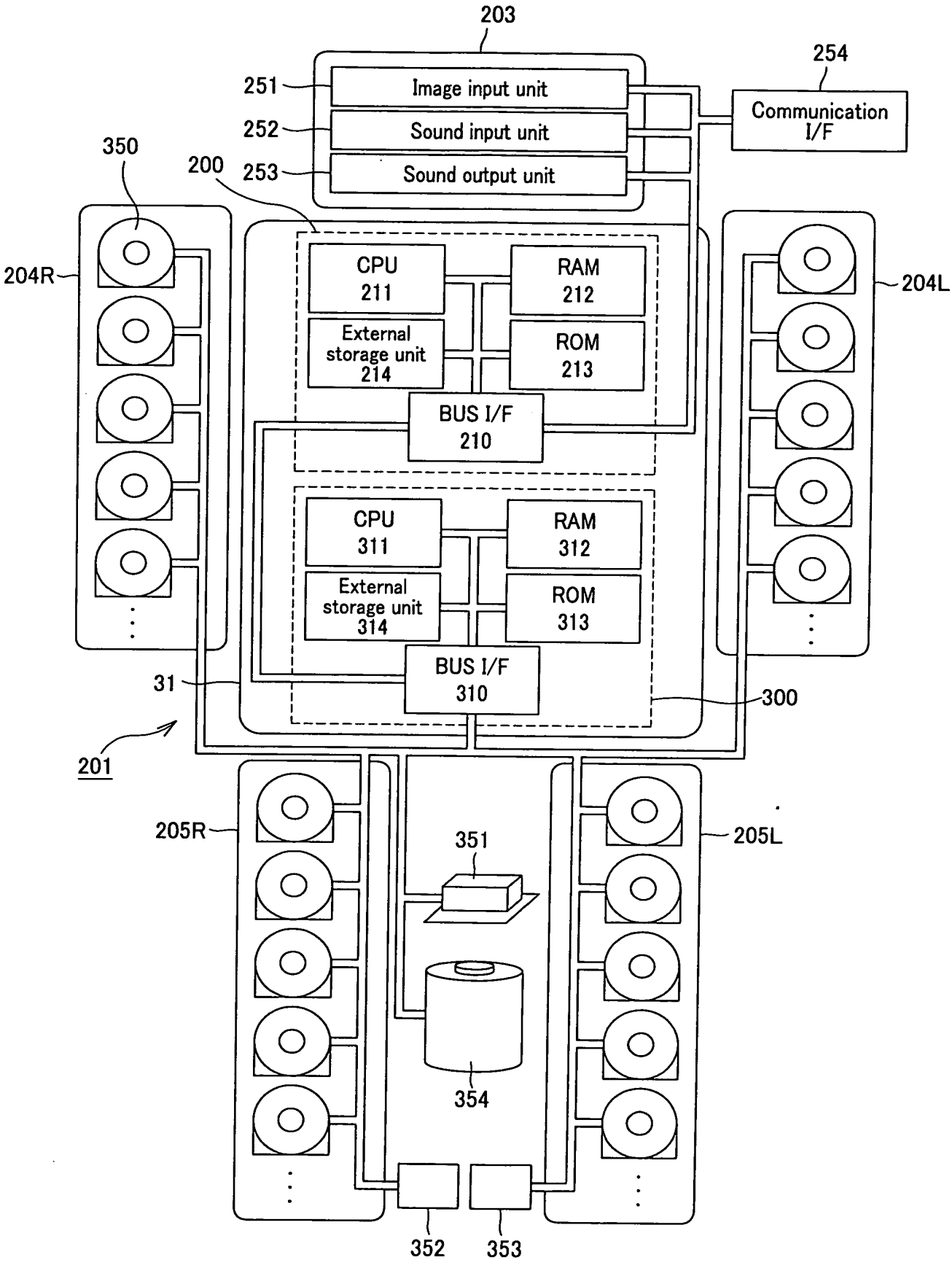

203
254
251
Image input unit
252
Sound input unit
253
Sound output unit
Communication I/F
350
200
204R
204L
CPU 211
RAM 212
External storage unit 214
ROM 213
BUS I/F 210
CPU 311
RAM 312
External storage unit 314
ROM 313
BUS I/F 310
31
300
201
205R
205L
351
354
352
353

FIG.9

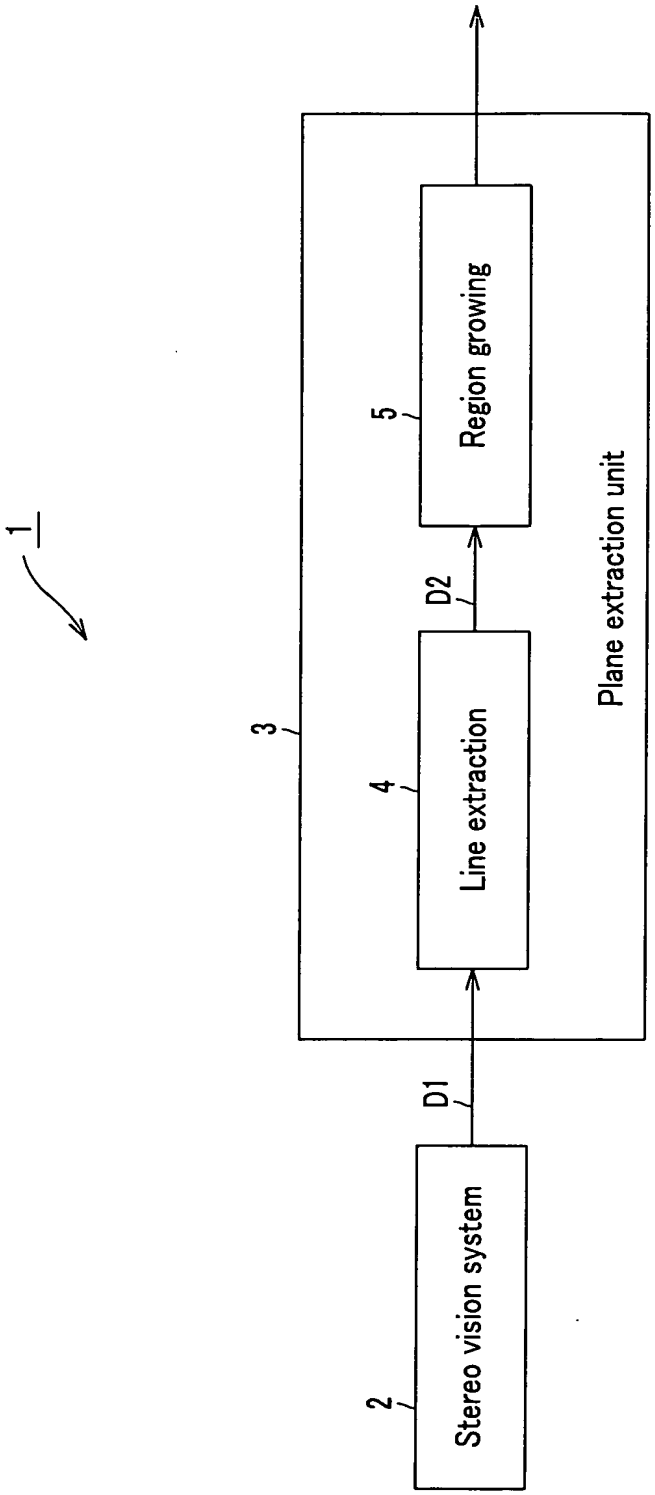

1
2
Stereo vision system
D1
3
4
Line extraction
D2
5
Region growing
Plane extraction unit

FIG.10

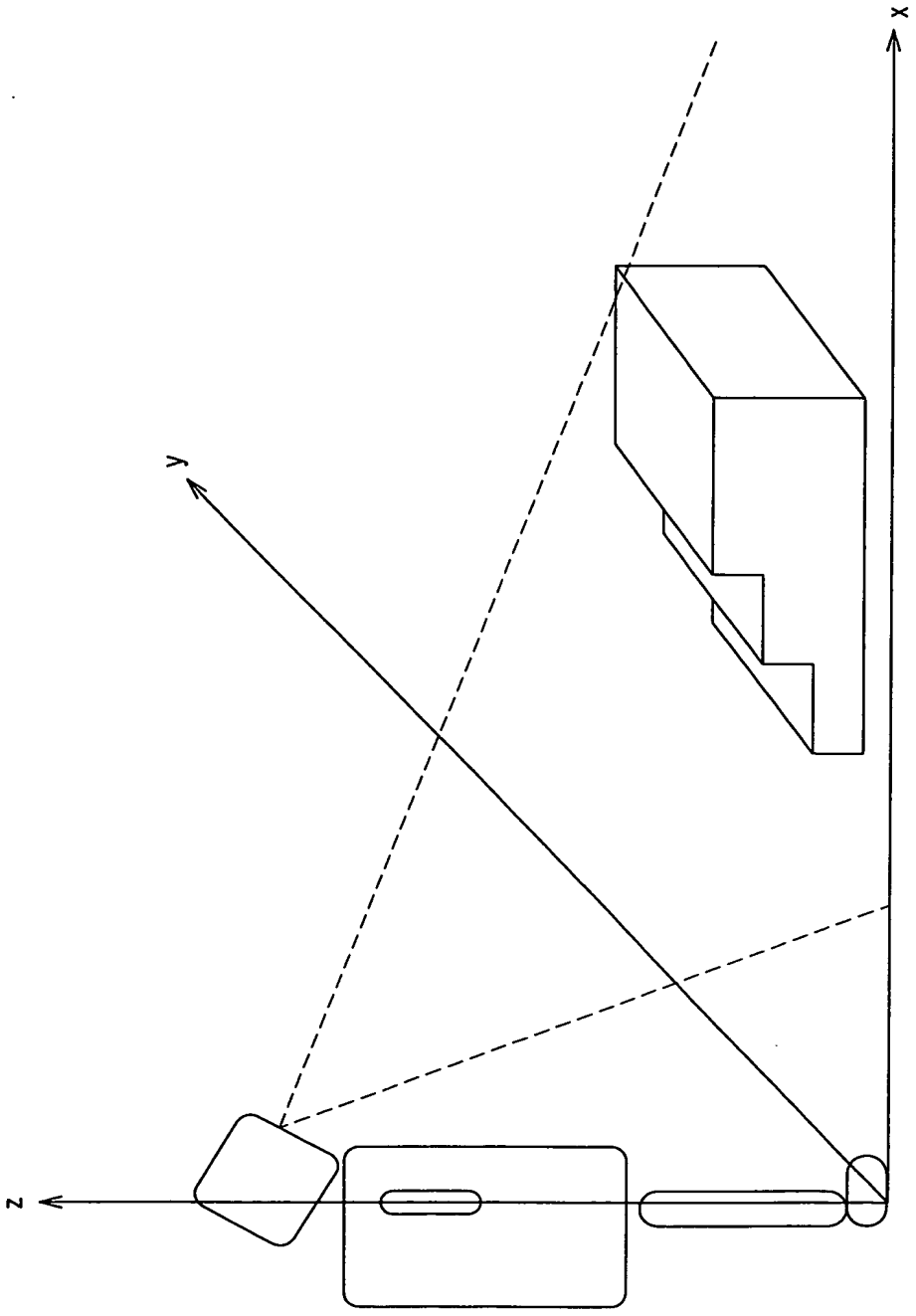



FIG.11

FIG.12A

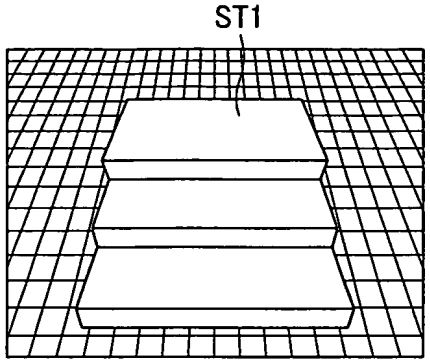



FIG.12B

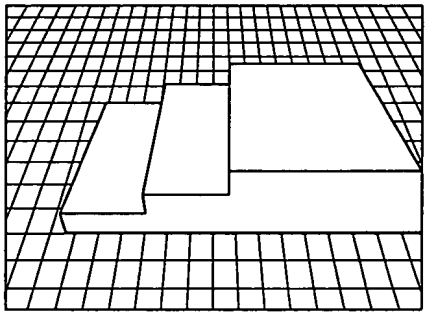
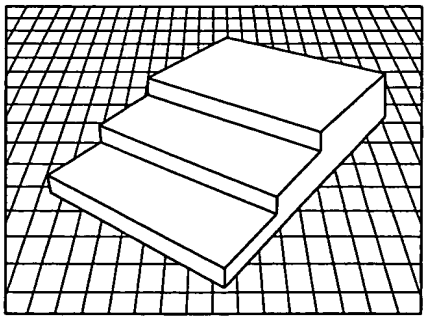


FIG.12C



4cm × 30cm × 10cm/21cm

FIG.13A

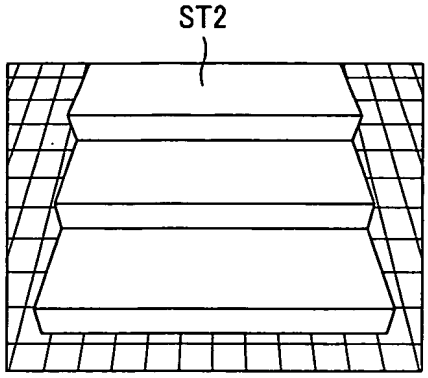



FIG.13B

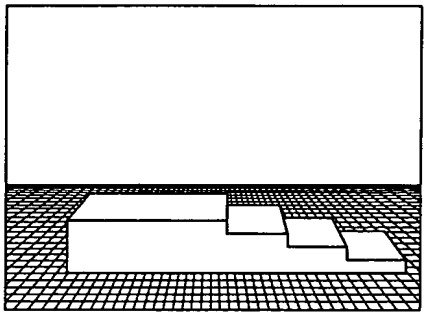
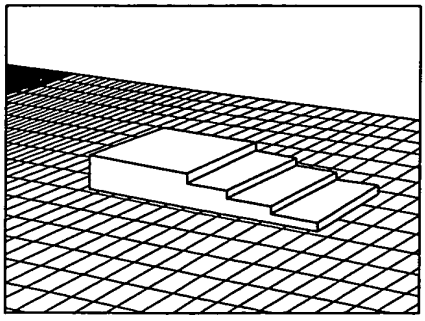


FIG.13C



3cm × 33cm × 12cm/32cm

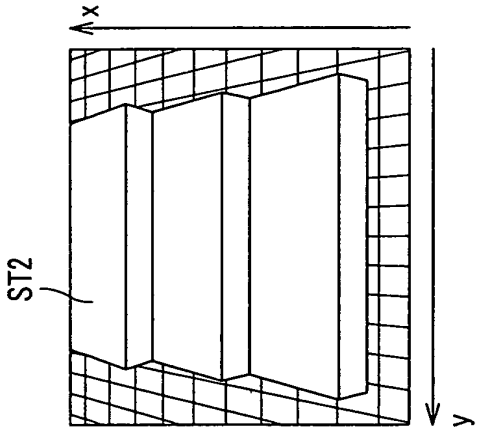



FIG.14A

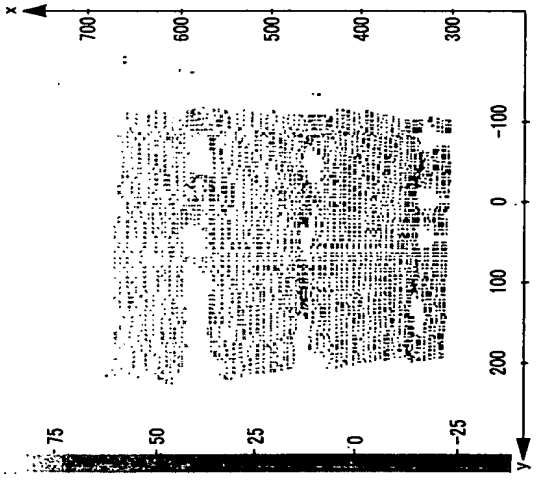



FIG.14B

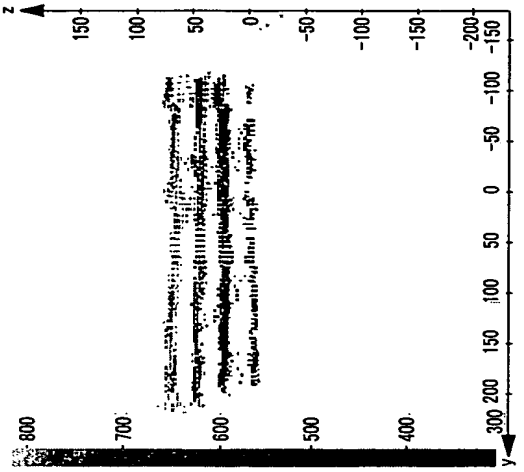



FIG.14C

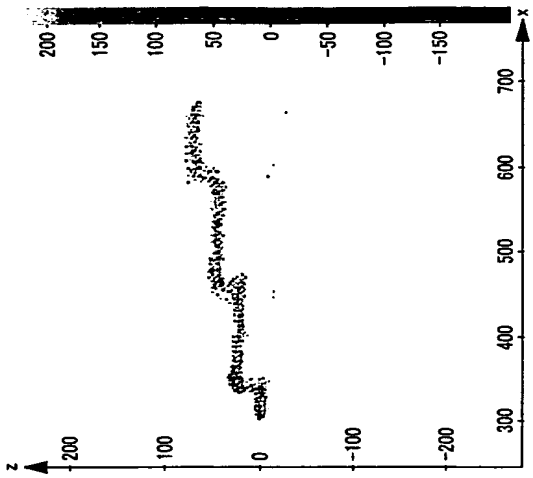



FIG.14D

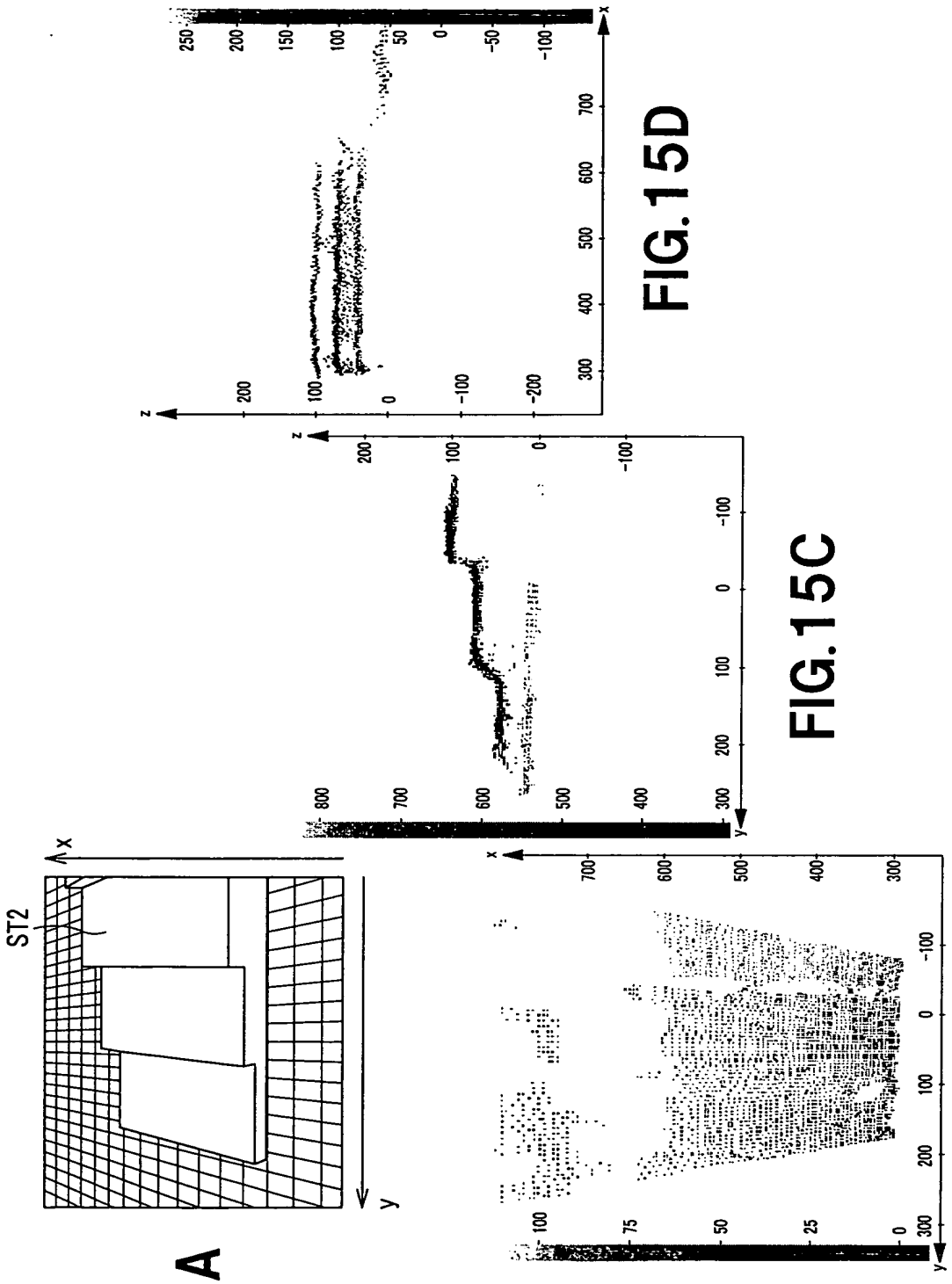



FIG.15A

FIG.15B

FIG.15C

FIG.15D

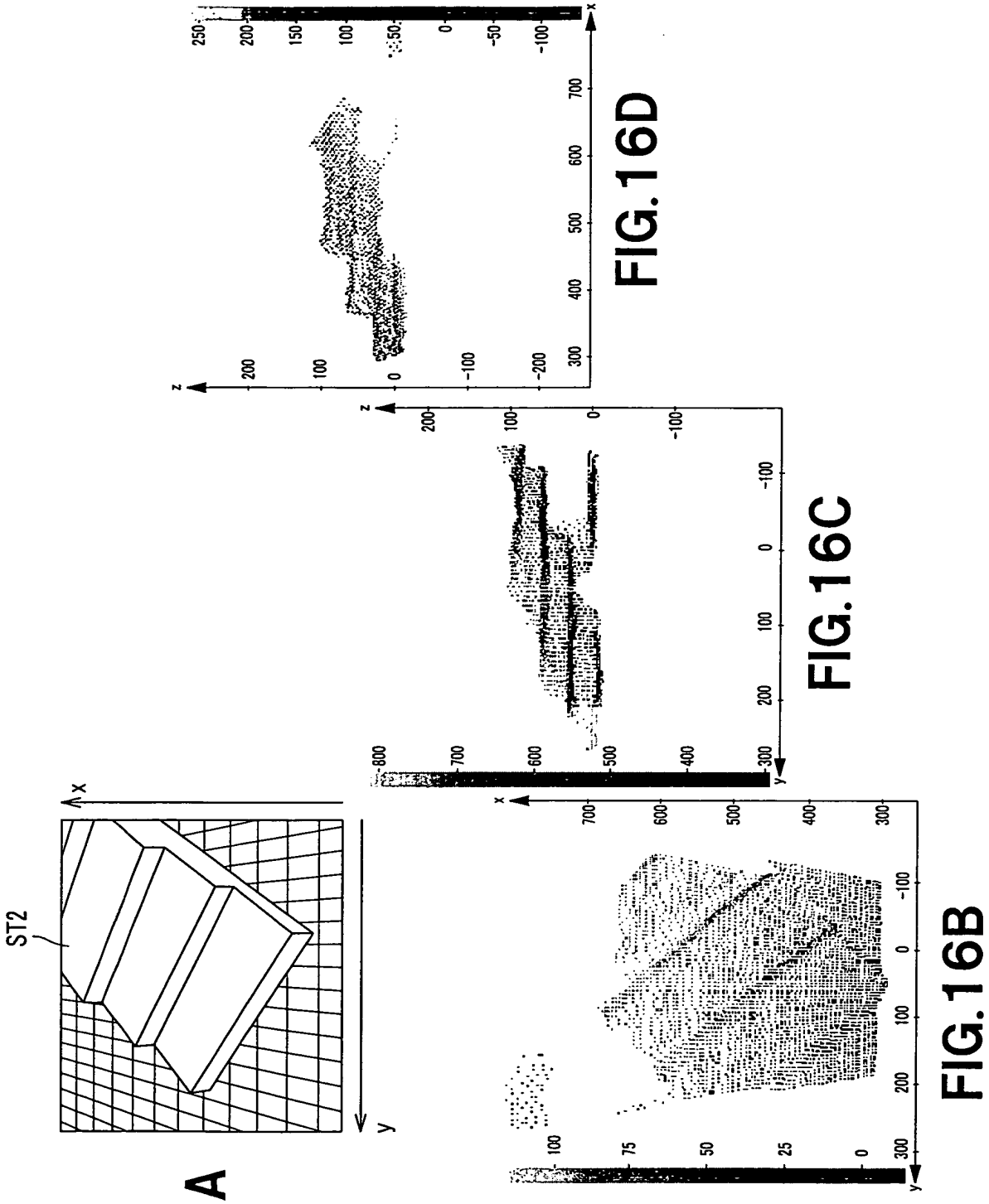



FIG.16A

FIG.16B

FIG.16C

FIG.16D

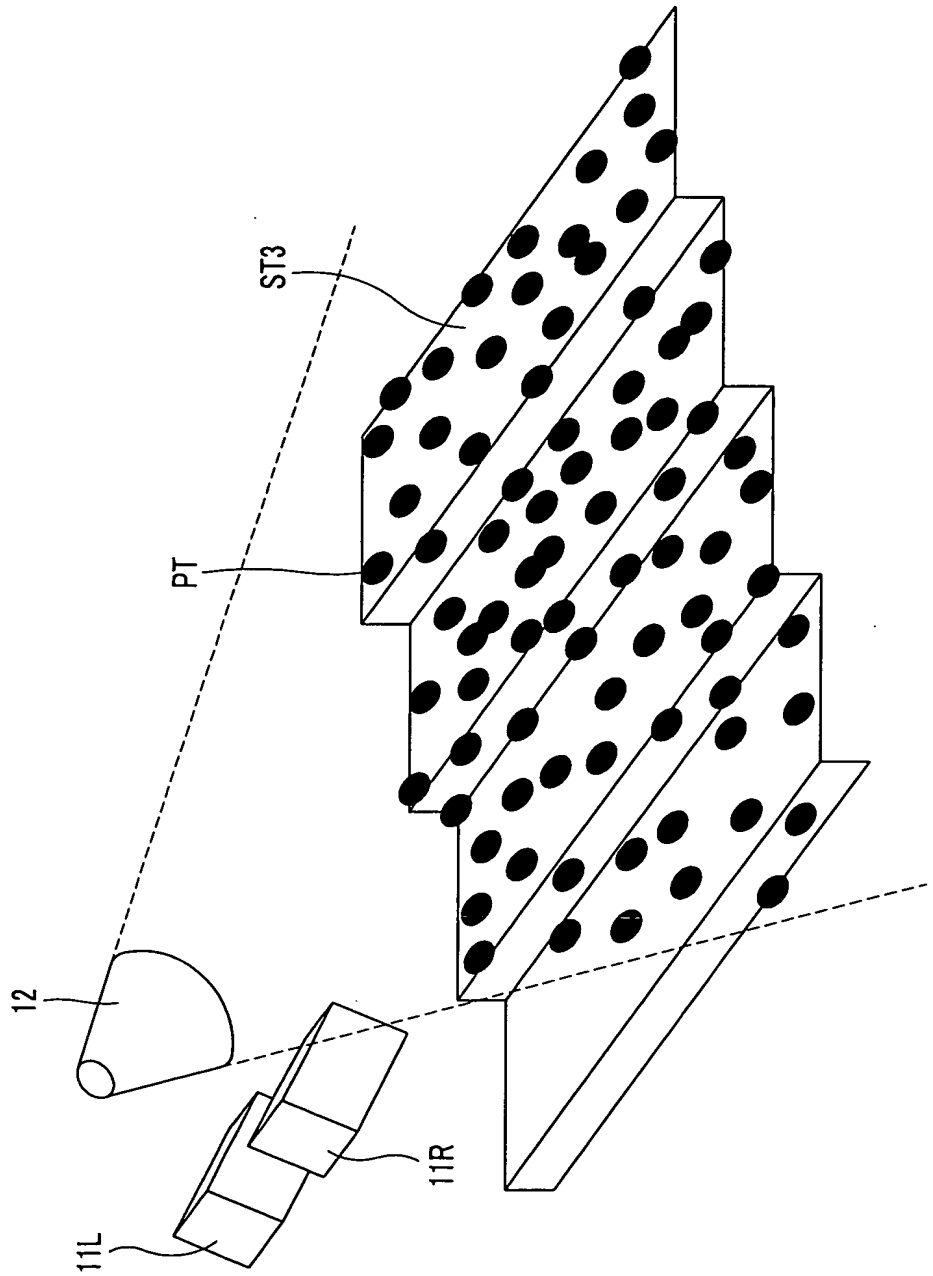



FIG.17

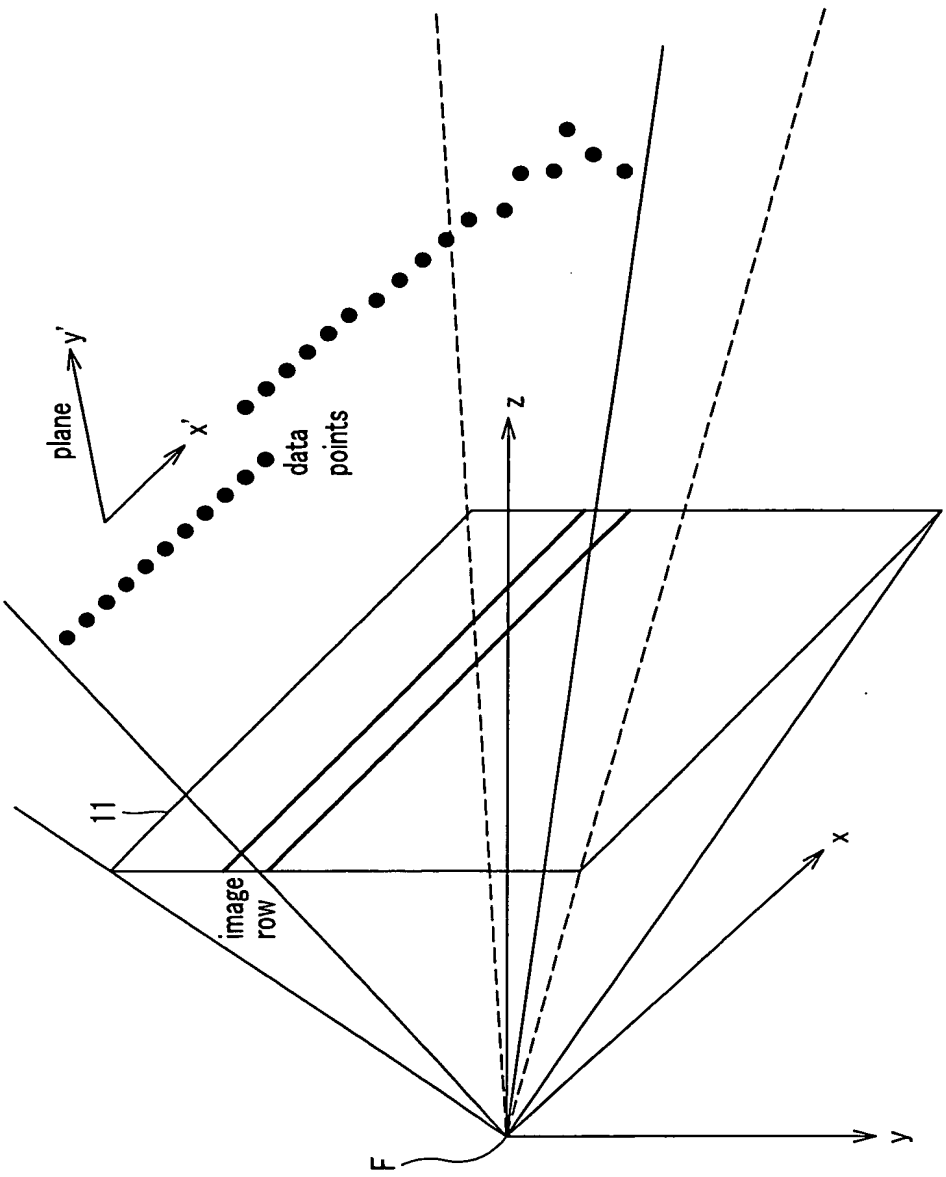



FIG.18

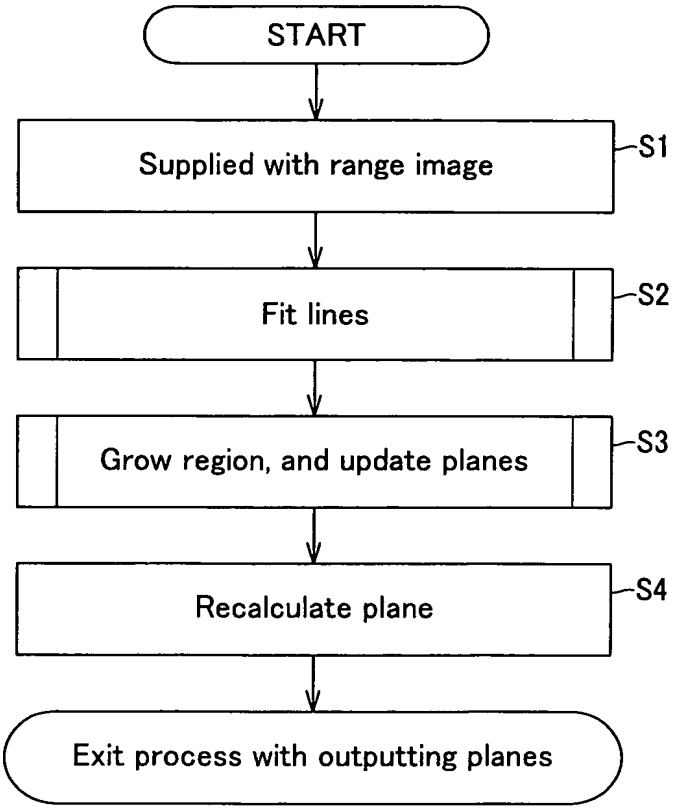

START
Supplied with range image
S1
Fit lines
S2
Grow region, and update planes
S3
Recalculate plane
S4
Exit process with outputting planes

FIG.19

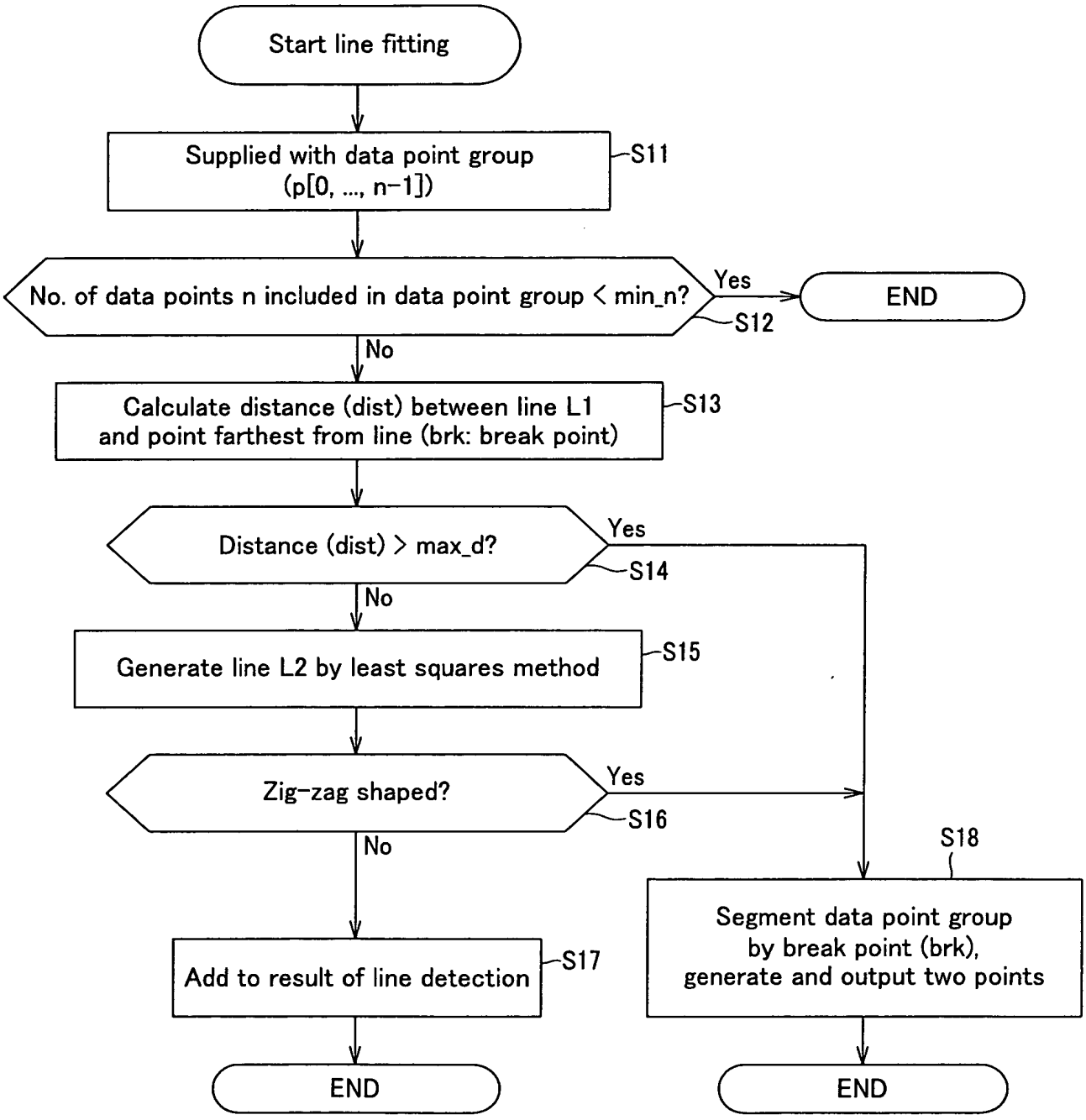
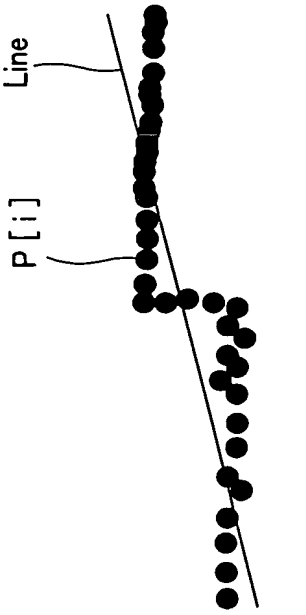
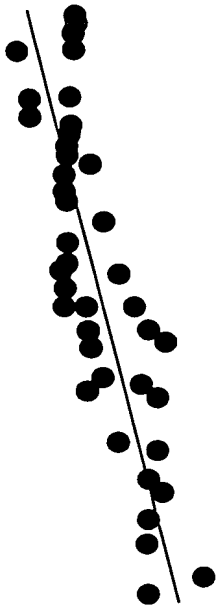

Start line fitting
Supplied with data point group (p[0, ..., n-1])
S11
No. of data points n included in data point group < min_n?
Yes
END
S12
No
Calculate distance (dist) between line L1 and point farthest from line (brk: break point)
S13
Distance (dist) > max_d?
Yes
S14
No
Generate line L2 by least squares method
S15
Zig-zag shaped?
Yes
S16
No
S18
Segment data point group by break point (brk), generate and output two points
Add to result of line detection
S17
END
END

FIG.20




zig-zag-shape

FIG.21A



random

FIG.21B

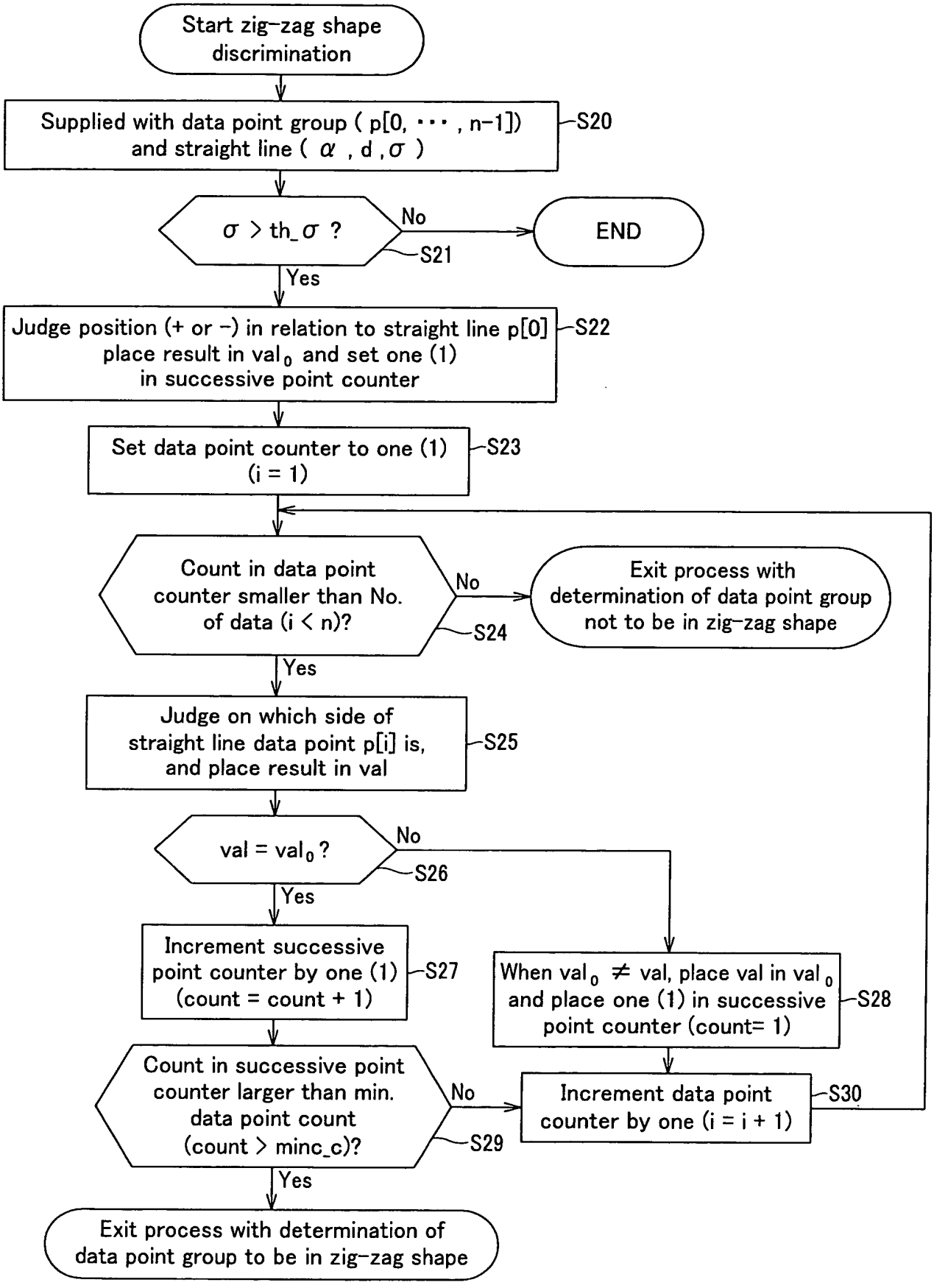

Start zig-zag shape discrimination
Supplied with data point group ( p[0, ・・・ , n-1]) and straight line ( α , d , σ )
S20
σ > th_σ ?
No
END
S21
Yes
Judge position (+ or −) in relation to straight line p[0] place result in val0 and set one (1) in successive point counter
S22
Set data point counter to one (1) (i = 1)
S23
Count in data point counter smaller than No. of data (i < n)?
No
Exit process with determination of data point group not to be in zig-zag shape
S24
Yes
Judge on which side of straight line data point p[i] is, and place result in val
S25
val = val0 ?
No
S26
Yes
Increment successive point counter by one (1) (count = count + 1)
S27
When val0 ≠ val, place val in val0 and place one (1) in successive point counter (count= 1)
S28
Count in successive point counter larger than min. data point count (count > minc_c)?
No
Increment data point counter by one (i = i + 1)
S30
S29
Yes
Exit process with determination of data point group to be in zig-zag shape

FIG.22

Input: *pts* : vector of points, *n* : number of points, $\alpha, d, \sigma$ : parameters and std-dev of fitted line.

Output : true if curve contains a *zig-zag*- shape, false otherwise.

Sequence :

```
if σ > threshσ  then
val0 = pts[0].x* cos α + pts[0].y* sin α + d
count = 1
  for i = 1 to (n-1) do
      val = pts[i].x* cos α + pts[i].y* sin α + d
      if (val* val0 <=0) then
          val0 = val
          count = 1
      else
          count = count + 1
          if (count >= min-points-for-zig-zag-shape ) then
              return true
          endif
      endif
  endfor
endif
return false
```

FIG.23

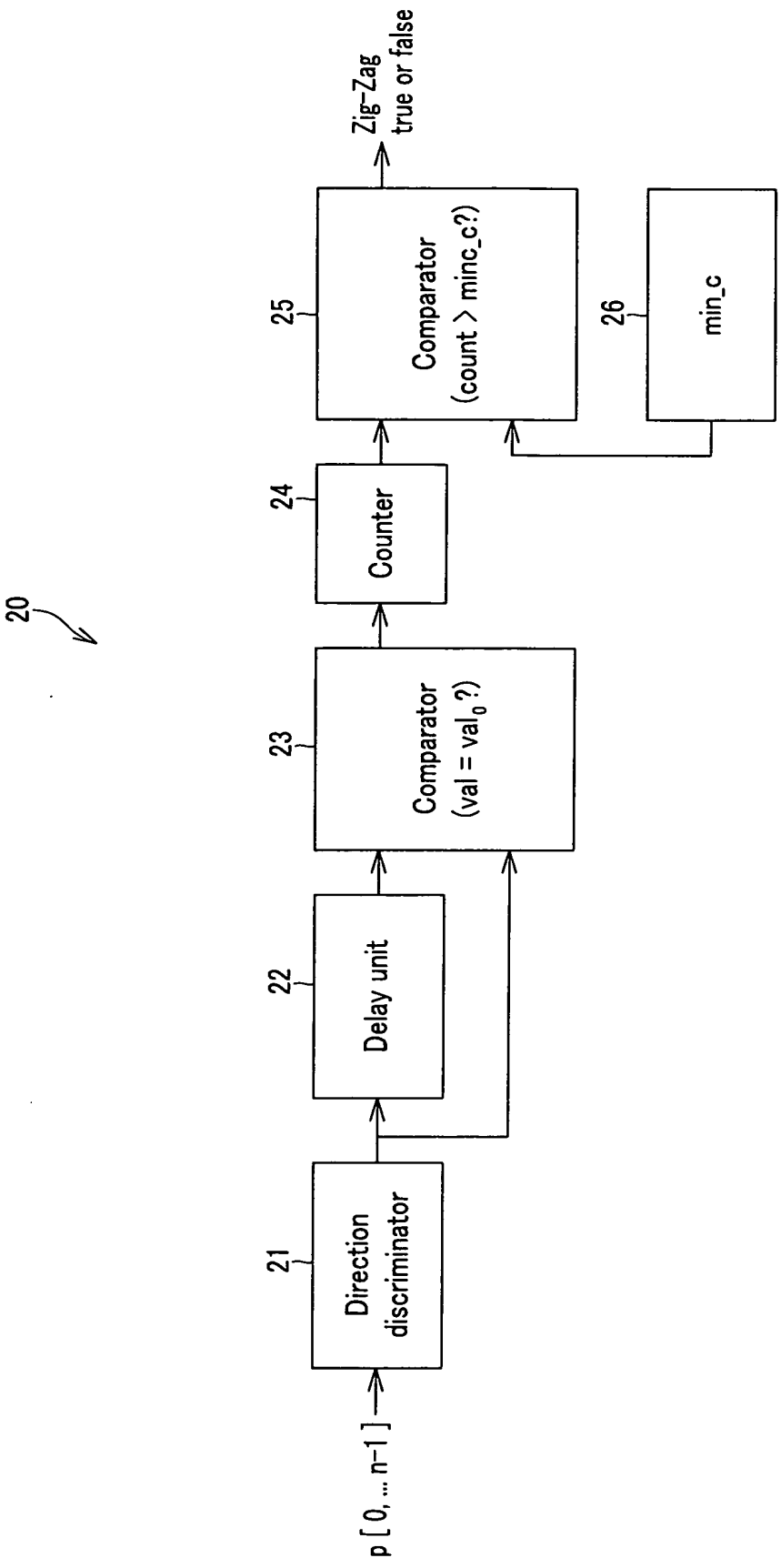

20
p [ 0, ... n-1 ]
21
Direction discriminator
22
Delay unit
23
Comparator (val = val0 ?)
24
Counter
25
Comparator (count > minc_c?)
26
min_c
Zig-Zag true or false

FIG.24

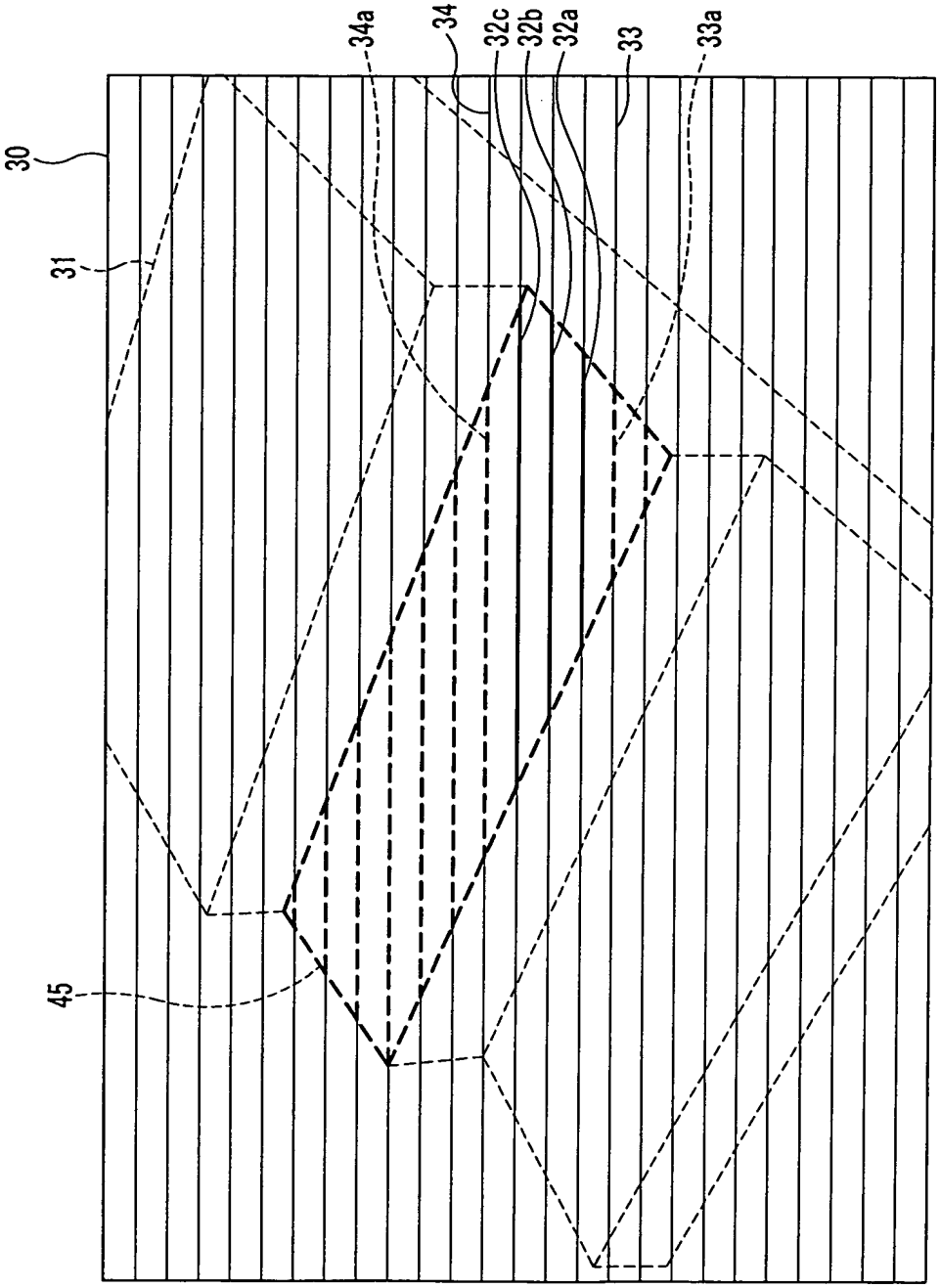



FIG.25

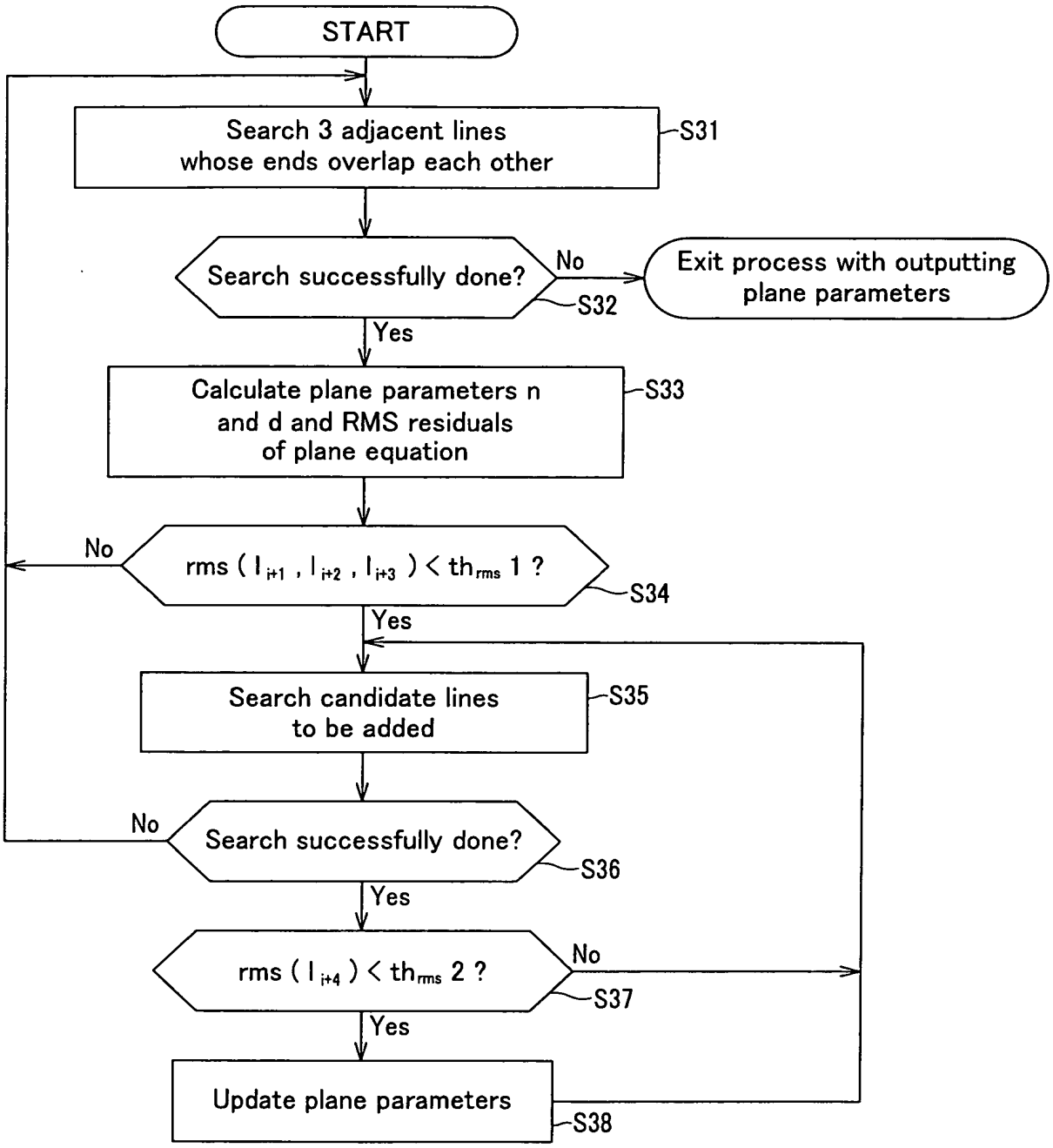

START
Search 3 adjacent lines whose ends overlap each other
S31
Search successfully done?
No
S32
Exit process with outputting plane parameters
Yes
Calculate plane parameters n and d and RMS residuals of plane equation
S33
No
rms ( I i+1 , I i+2 , I i+3 ) < th rms 1 ?
S34
Yes
Search candidate lines to be added
S35
No
Search successfully done?
S36
Yes
rms ( I i+4 ) < th rms 2 ?
No
S37
Yes
Update plane parameters
S38

FIG.26

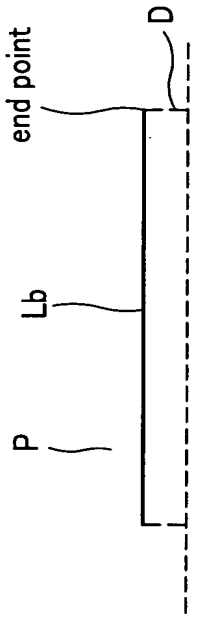



FIG.27B

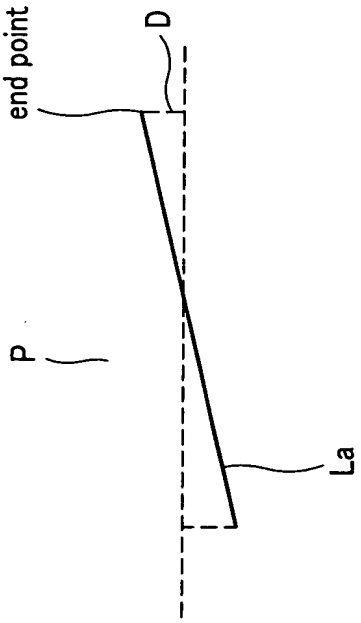



FIG.27A

Algorithm *FindSeedRegion*

Input: *lines[i]*: vector of lines for esch image row (or column) $i$,

$n$: number of image rows (or columns)

Output : set of lines (seed region) or empty set (no seed found).

Sequence :

```
for i = 0 to (n-3) do
    for l1 in lines[i] do
        for l2 in lines[i+1] do
            for l3 in lines[i+2] do
                if overlap(l1 , l2 ) and overlap(l2 , l3 ) then
                    (n,d) = fitPlane(l1 , l2 , l3 )
                    if rms(l1 , l2 , l3 ) < thresh1rms then
                        seed = {l1 , l2 , l3 }
                        remove(l1 , l2 , l3 )
                        return seed
                    endif
                endif
            endfor
        endfor
    endfor
endfor
return {}
```

FIG.28

Algorithm *RegionGrowing*

Input: *region* : set of lines as seed region,

*lines[i]*: vector of lines for esch image row (or column) $i$,

$n$: number of image rows (or columns)

Sequence :

*A = 0, b = 0*

for *1 in region* do *(A,b) = add(A,b,l)* endfor

*(n,d) = solve(A,b)*

*open = region*

while *not empty(open)* do

$1_1$ *= select(open),open = open - {*$1_1$*}*

for *i in neighbor(index(*$1_1$*))* do

for $1_2$ *in lines[i]* do

if *overlap(*$1_1$*,*$1_2$*)* and *rms(*$1_2$*) <* $thresh2_{rms}$ then

*region = region + {*$1_2$*},*

*(A,b) = add(A,b,*$1_2$*), (n,d) = solve(A,b)*

*open = open + {*$1_2$*},*

*remove(*$1_2$*)*

endif

endfor

endfor

endfor

*plane = {n,d,A,b,region}*

*planes = planes + {plane}*

FIG.29

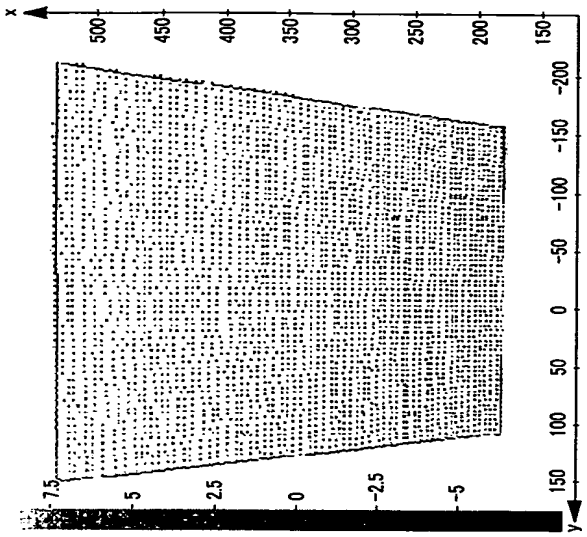



FIG.30C

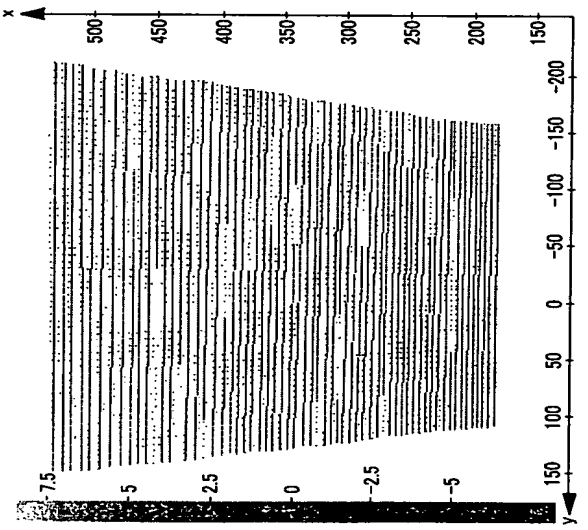



FIG.30B

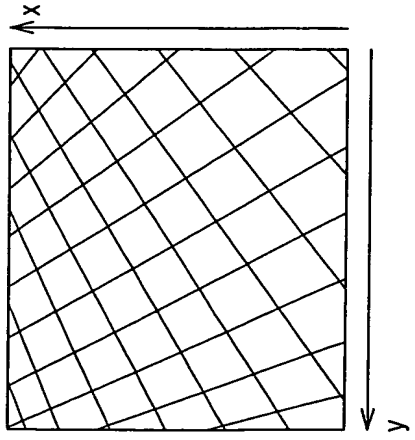



FIG.30A

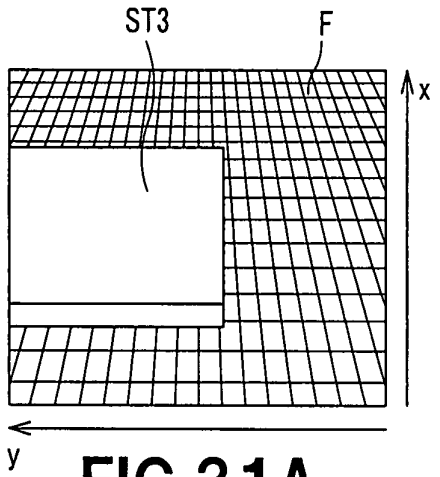



FIG.31A

| No | max_d | enable zig-zag | correct extraction (horizontal) | correct extraction (vertical) |
|---|---|---|---|---|
| 1 | 30 | no | 0 / 10 | 0 / 10 |
| 2 | 25 | no | 0 / 10 | 0 / 10 |
| 3 | 20 | no | 10 / 10 | 0 / 10 |
| 4 | 15 | no | 10 / 10 | 3 / 10 |
| 5 | 10 | no | 10 / 10 | 10 / 10 |
| 6 | 30 | yes | 10 / 10 | 10 / 10 |

FIG.31B

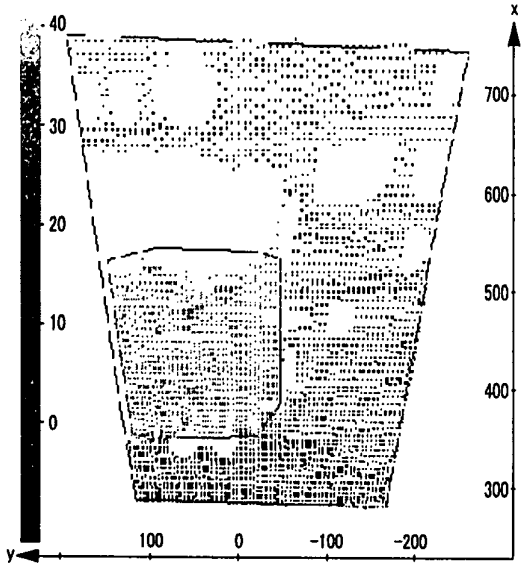



FIG.31C

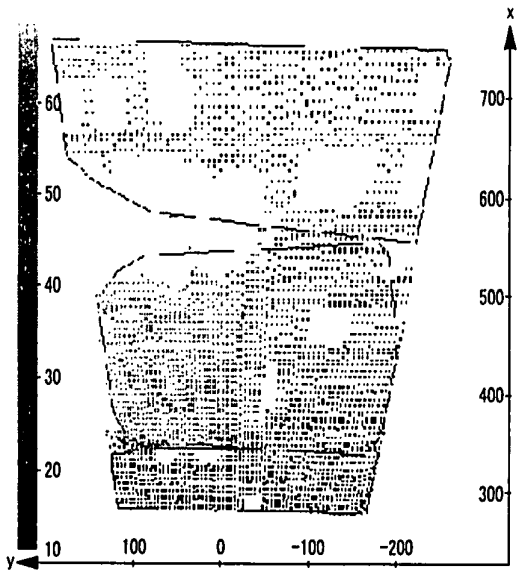



FIG.31D

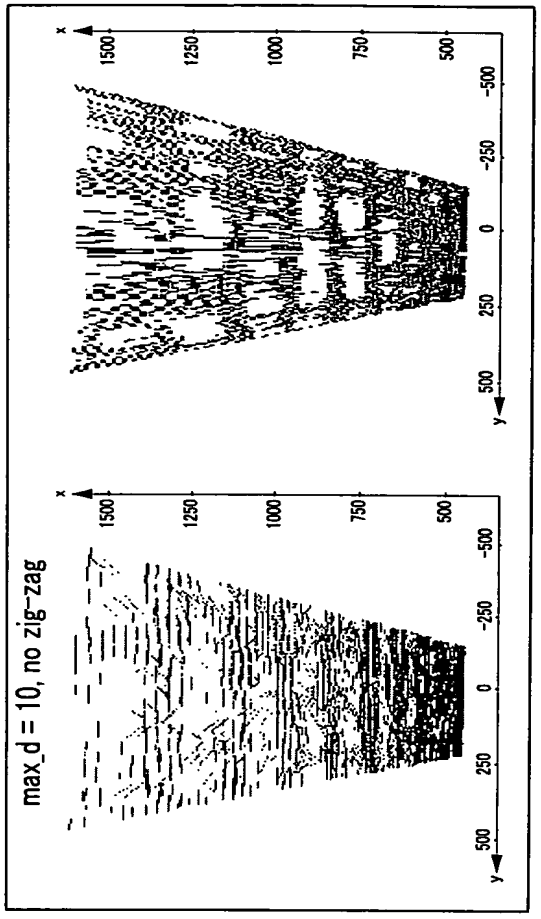



FIG.32B

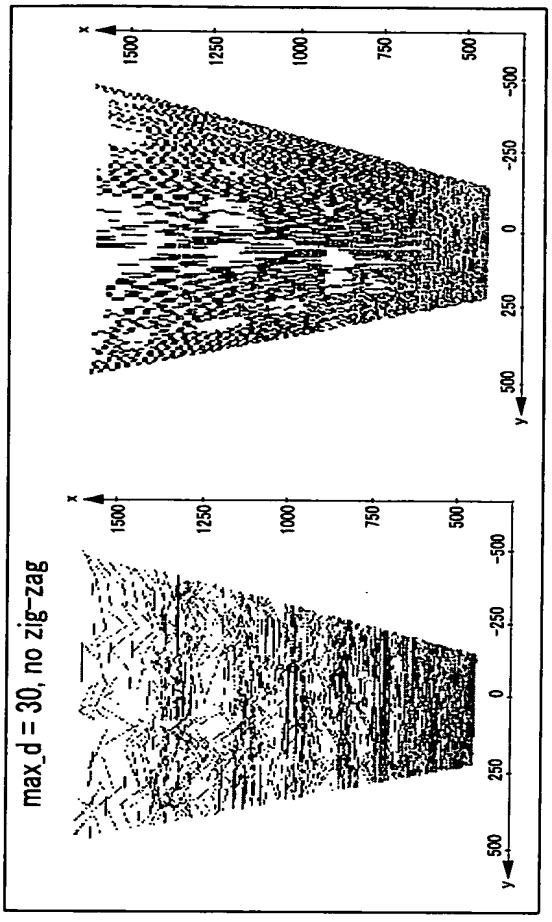



FIG.32C

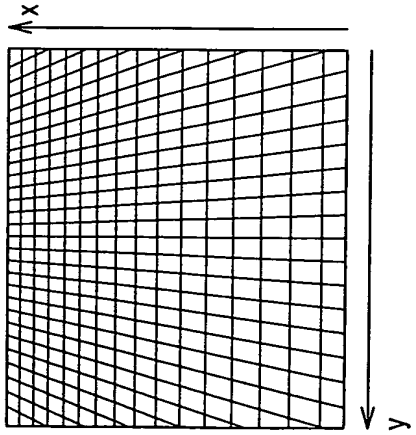



FIG.32A

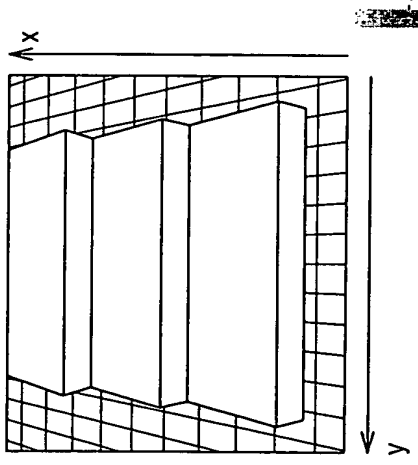



FIG.33A

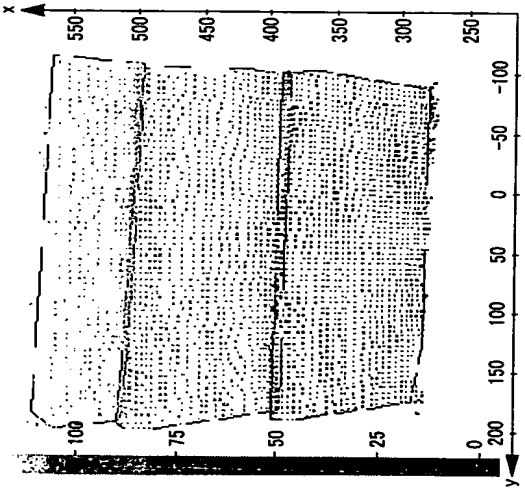



FIG.33B

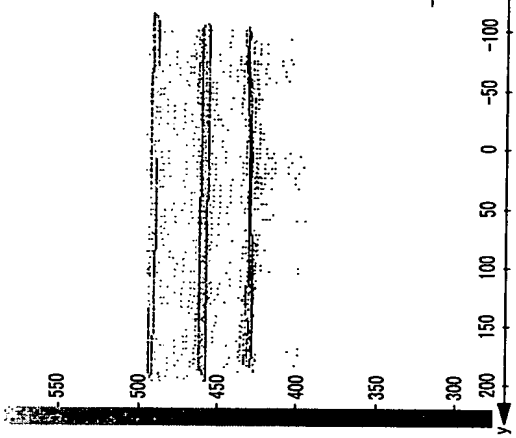



FIG.33C

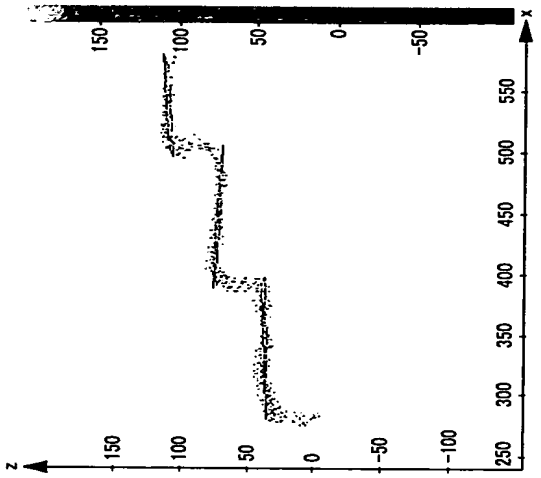



FIG.33D

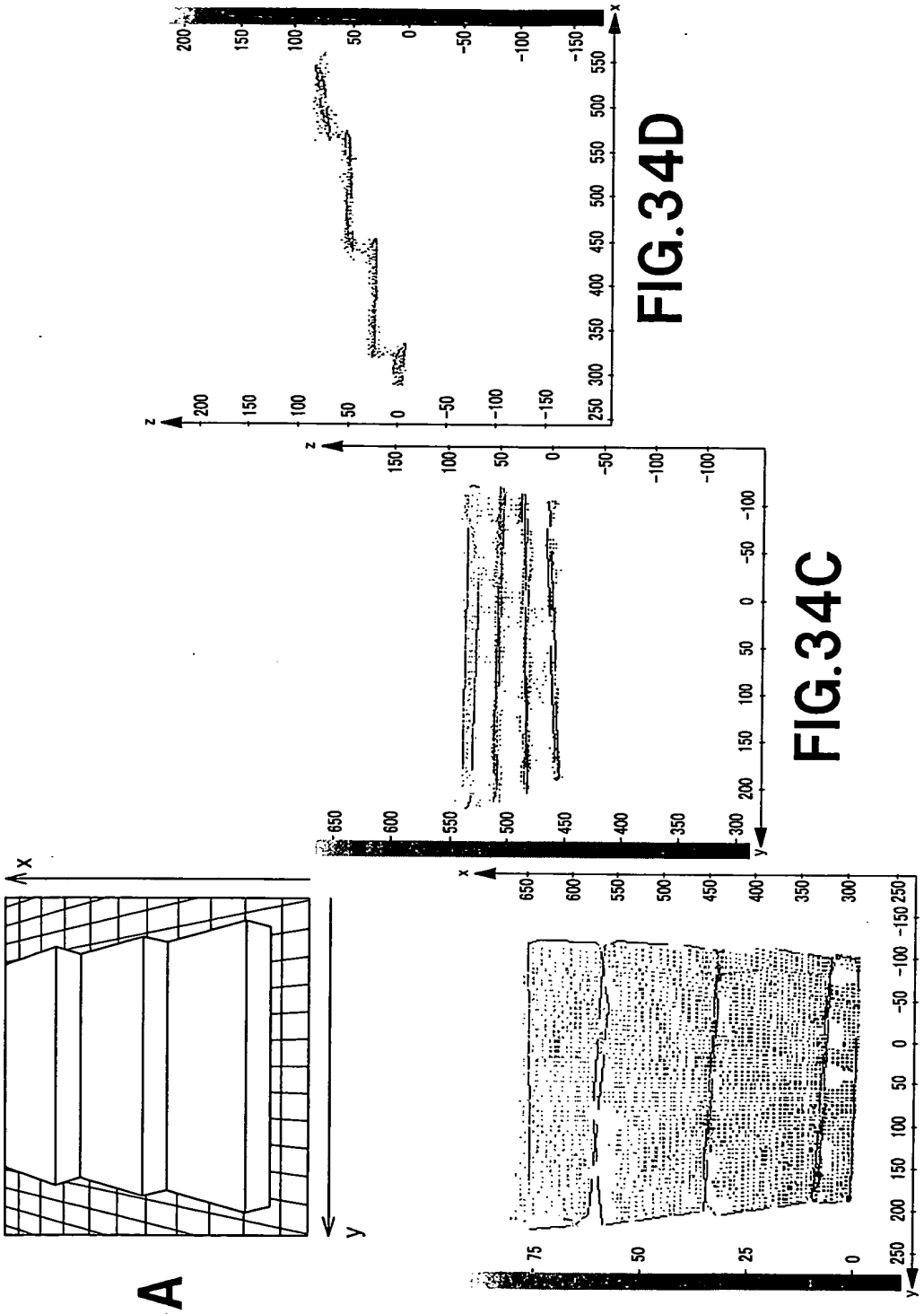



FIG.34A

FIG.34B

FIG.34C

FIG.34D